Example:

Creating built in attr_accessor method

In this task, we will create own method that behaves just like “attr_accessor” method from the Standard Library. Also, this will help you to demystify all the magic behind Rails’ “validates”, “belongs_to”, “find_by_*” and similar methods.

Task:

Create your own method that behave just like built-in “attr_accessor” method from Standard Ruby Library.

Name your helper method as “attr_accessible”:

```
attr_accessible :foo, :bar
```

This helper method should be accessible within all classes.

Background

```ruby
class MyClass
    attr_accessor :foo, :bar
end


obj = MyClass.new
obj.foo = "value of foo"
obj.foo                                 # "value of foo"
obj.bar                                 # nil

obj.instance_variables                  # [:@foo]

MyClass.instance_methods(false)         # [:foo, :foo=, :bar, :bar=]
```

Define method in Kernel module

```ruby
module Kernel
    def attr_accessible(*args)
        args.each do |name|
            define_method(name) { instance_variable_get("@#{name}") }
            define_method("#{name}=") { |new_value| instance_variable_set("@#{name}", new_value) }
        end
    end

    private :attr_accessible                    # This prevents: `MyClass.attr_accessible :foo`
end


class MyClass
    attr_accessible :foo, :bar
end

MyClass.instance_methods(false)                 # [:foo, :foo=, :bar, :bar=]

obj = MyClass.new
obj.foo = "value of foo"
obj.foo                                         # "value of foo"
obj.bar = "value of bar"
obj.bar                                         # "value of bar"
```

This should demystify Rails':

- attr_accessible
- find_by_*
- belongs_to
- etc.

Task:

Put this technique into practice by:

1. Implementing other Rails methos. Such as:
   - `validates`
   - `belongs_to`
   - etc.
2. Real life case from your own projects.

Limiting access to your helper methods

```ruby
class Parent
    private
    def self.attribute_reader(*args)
        args.each do |name|
            define_method(name) { instance_variable_get("@#{name}") }
        end
    end
end

class Child < Parent
    attribute_reader :foo, :bar
end

class MyClass
    attribute_reader :foo                    # NoMethodError: undefined method `attribute_reader' for MyC
end

Child.instance_methods(false)                # [:foo, :bar]
```